Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

A SIMULATION-BASED FINITE CAPACITY SCHEDULING SYSTEM

Alexander J Weintraub
Andrew Zozom, Jr.
Thom J. Hodgson
Denis Cormier

Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906, U.S.A.

ABSTRACT

This paper describes the methodology of a finite capacity scheduling system which uses an object-oriented discrete event simulation as its engine. The scheduling system is developed as an add-on to an infinite capacity MRP system. The system was developed in partnership with a high-end case goods furniture company. The schedules generated by the simulation-based system will be used to schedule all operations and personnel on the shop floor. The simulation methodology and statistics captured are described, along with industrial results. Results for some randomly generated problems are also given.

1 INTRODUCTION

1.1 Motivation for Model Development

In recent years, several developments have significantly changed the way products are produced. The emergence of a global marketplace has resulted in new competition along with increased pressure to provide more features, customization, and quality for products all at lower prices. On the shop floor, the combined effects of these developments can be seen in the form of custom orders and very small lot sizes as companies try to avoid the risk of overnight product obsolescence. In order to avoid this risk, companies have abandoned large lot build-to-stock strategies in favor of small lot built-to-order strategies. Problems arise when the assumptions of the production scheduling system do not match this new strategy.

Traditional MRP-based systems assume infinite machine capacities. The result is that at specific points in time in the scheduled plan, certain machine centers become overloaded. Floating bottlenecks can occur at the various machine centers, which prevents timely movement of parts through a plant.

The MRP system of our industrial partner uses a one day buffer between each operation as an estimate of material handling and queuing time between machines. Queuing on a previous machine can affect the arrival time of a part to the next machine center. With an infinite capacity model, it is assumed that all parts in queue on a given day will be processed. The infinite-capacity MRP model is therefore not accurate enough to properly generate a schedule.

A desired capability of any scheduling system is to be able to look into the future. The company described in this paper maintains a database on the current location of every part (lot) in the work-in-process inventory. The procedure described here provides a vehicle by which a manager can look into the future in terms of optimized system performance. As a side benefit, the solution provides priorities for sequencing jobs at every machine. Since the system will operate in essentially real-time, alternative resource management schemes (overtime, re-assignment of manpower, maintenance schedules, etc.) can be evaluated. This will allow management to play out alternatives realistically and rapidly to determine which of several courses of action is most appropriate. A finite capacity model eliminates the MRP shortcomings mentioned above and produces a schedule that can be met on the shop floor.

A simulation-based approach to scheduling is described which provides optimal, or near optimal, schedules for the N -job, M -machine, maximum lateness problem ($N/M/L_{\max}$). This "Virtual Factory" is a discrete-time deterministic simulation of a large scale job shop which uses real-time MRP-based data (e.g., process times, setup times, location of parts, process plans) to build a model of the factory shop floor.

1.2 Manufacturing Process

A typical furniture manufacturing process consists of six stages: drying, rough cutting, machining, assembly, finishing (or lacquering), clean up, and then stor-

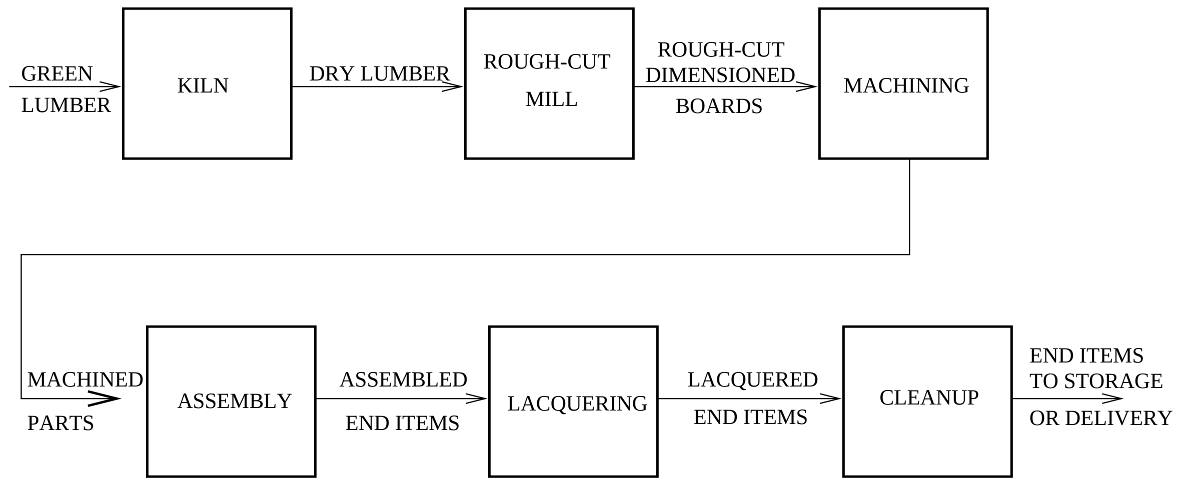


Figure 1: Typical Furniture Manufacturing Process

age or shipping, depending on the disposition of the lot. (See Figure 1)

The first stage, lumber drying, takes place very early in the production timeline. Prior to entry into the factory, the raw green lumber to be used is treated. This treatment includes a drying process which takes place in large kilns. In order to maximize the yield of the ovens, the batch sizes are large (ranging from 20,000 to 100,000 board-feet) and each batch usually is made up of the same species of wood. The large batches of dried wood are then delivered to the factory.

The second stage is the cutting of the dried rough lumber to size in the rough-cut mill. The rough-cut mill cuts the lumber into unfinished, dimensioned boards based on the production plan requirements for the cutting. The output from a batch is given an identification number and the entire batch is referred to as a cutting. Each part is identified and bar-coded. The part and cutting numbers are used to track the flow of materials through the plant.

From the rough-cut mill the raw parts go to the third stage, machining. The machine room can have many different types of machines to prepare the parts for final assembly. Sanding, milling, drilling and detailed shaping are some of the operations done in the machine room. The routing of each board has been predetermined and depends on the machining requirements of each particular piece of furniture. (For example, a more detailed table leg may be processed by a different set of machines than a plain one.)

The fourth stage is assembly. All end-items in a cutting are scheduled to arrive at the assembly lines around the same time. Since the raw materials are homogeneous for the entire cutting, furniture pieces of a similar style are machined and assembled concurrently. (Similarly styled pieces comprise a suite. An example of a suite might be a collection of oriental-style black lacquered furniture.)

At assembly the parts are grouped by end-item. (An end-item is a piece of furniture.) The end-item is then put together on one of several assembly lines. Glass pieces, drawers, knobs and other hardware are also put on during assembly. The end-items from a single cutting may occupy the assembly lines for five to ten days. This period of time is referred to as the suite's bucket. The assembled end-items then travel to lacquering, and then to final cleanup and inspection.

While there are variations in the manufacturing process described above from plant to plant, it is typical of high-end case goods (e.g., tables and bureaus) manufacturing.

1.3 Current Scheduling Process

A typical scheduling timeline is as follows: Four to six months in advance, demand forecasts are used to determine which products (i.e., which end-items within a suite) and roughly how many of each item in the suite should be manufactured. A bucket is then tentatively scheduled for that suite. Due dates for end-items within the bucket are not finalized at this time.

Approximately two months in advance of assembly, the sequence of end-items in the bucket is finalized. This sequence is fixed at this time. The result of this assignment is a set of due dates for parts and subassemblies to arrive at the assembly line for each end-item. These due dates then are projected back through the factory, using the MRP database information, to determine the planned movement of material through the plant.

2 SLACK-BASED PRIORITY RULE

Minimizing the maximum lateness ($L_{\max}$) of any part was agreed upon as the appropriate objective. In the case described here, the "customer" is the final assembly operation. If $L_{\max} > 0$, production schedules in final assembly would be delayed. The minimizing of a measure such as average tardiness or average lateness may lead some jobs to be very tardy and some jobs to be very early. Such a schedule would not be satisfactory since meeting customer due dates is the primary concern. Therefore, due date ordering was an obvious first choice for determining the job sequences at each machine.

For each job j, there is a specified final due date, d_j. To translate this final due date, d_j, to a local due date, d_{jk}, where k is the operation index of job j, an estimate of the remaining time required to process all operations from k to the final operation for job j is required.

An approach that has been used to determine due dates at intermediate machines is to calculate an effective due date (called *slack*) for each job j at each machine m. The quantity *slack* takes into account processing times subsequent to the machine in question:

$$slack_{j,k} = d_j - \sum_{k \in m_k^+} p_{jk}$$

where m_k^+ is the set of all subsequent operations to machine m on the routing sheet. The quantity $slack_{jk}$ represents the latest time that the job can finish processing on machine m and still satisfy its final due date.

If each job is processed in order of ascending *slack* at each machine m, then intuitively it seems that $L_{\max}$ would be minimized. One of the earliest mentions of *slack* in the literature appeared in (Rowe 1960). In early experimental tests (Carroll 1965) *slack* did not perform particularly well as a sequencing tool (*slack per remaining operation* performed somewhat better). One reason for this poor performance is that the *slack* calculation, as defined, does not take into account any queuing that may occur over the subsequent machines in a part's route.

The queuing time of a job has been shown to account for up to ninety percent of its remaining time in the production system (Plossl 1985). Thus, any scheduling heuristic for large job shops should include the remaining queuing time of a job in determining the sequence of jobs at a machine. An estimate of *slack* must include all remaining processing times of job j, as well as all queuing times at each remaining operation from k until completion and also any material handling times. This estimate has been referred to as the tail lead time, or TLT_{jk} (Morton and Pentico 1993). Thus, the local due date can be estimated as $d_{jk} = d_j - TLT_{jk}$.

There are three procedures that can be used to estimate the tail lead time: 1) an historical fixed parameter, 2) an historical varied parameter, or 3) an iterative estimation. An historical fixed parameter estimates the lead time as a multiple, w, of the remaining processing time. An historical varied parameter would vary the multiple value, w, based on certain parameters. These parameters can be the load of the shop or the size of the queue at the machine in question. One iterative procedure to determine an estimate of the tail lead time is described below (Morton and Pentico 1993).

Step 1: Choose any initial estimate of the lead times (e.g. $w = 3$).

Step 2: Perform a simulation of the job shop utilizing a priority dispatch heuristic.

Step 3: Record the objective function obtained, $F(n)$ for iteration n.

Step 4: Record actual lead times, $ATLT_{jk}(n)$ for iteration n.

Step 5: Make smoothed new estimates:

$$TLT_{jk}(n+1) = (1-\alpha)ATLT_{jk}(n) + \alpha TLT_{jk}(n)$$

Step 6: If the termination condition is satisfied, go to Step 8.

Step 7: Go to Step 2.

Step 8: Report the objective value for the iteration giving the best value of the objective function.

The authors report consistently good results using the iterative procedure. They state that the objective value improves quite fast during the first few iterations and then fluctuates due to the discreteness of the processing times. A termination condition in Step 6 could be stated as "stop if there is no improvement after ten iterations."

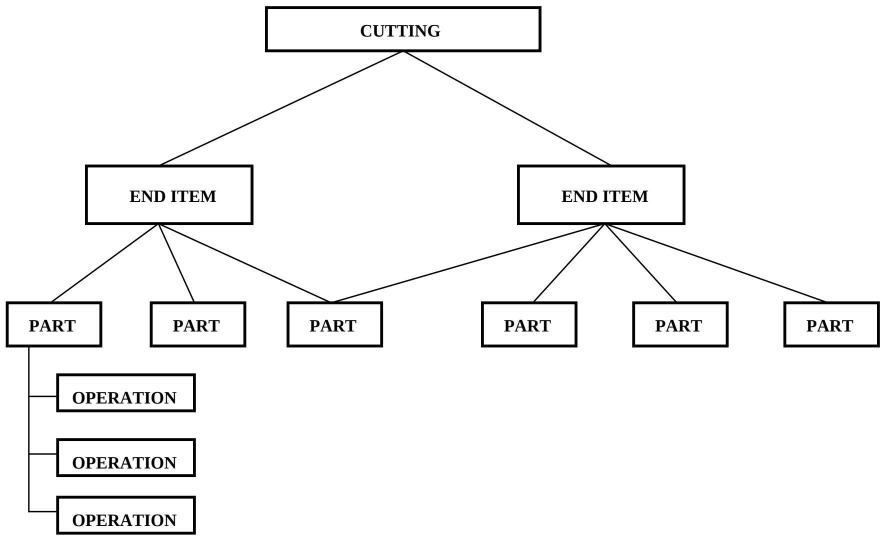


Figure 2: Conceptual Diagram of Class Hierarchies

A proposed revised slack ($slack'$) calculation to include this queuing time was investigated. The $slack'$ calculation is defined as:

$$slack'_{j,k} = d_j - \sum_{j \in m_k^+} p_{j,k} - \sum_{j \in m_k^{++}} q_{j,k}$$

where $\sum_{j \in m_k^{++}} q_{j,k}$ is the total queuing time for job j for all subsequent operations to machine m except the immediate subsequent operation. In other words, $slack'_{jk}$ is the time at which the job must finish processing in order to be available to start processing at its next operation. The effective due date at the subsequent machine center takes into account all subsequent queuing times as well. These subsequent queuing times are values obtained from the previous iteration of the job shop simulation.

3 SIMULATION DESCRIPTION

3.1 Modeling Process

This Virtual Factory model is designed as an add-on to a Materials Requirements Planning (MRP) system. The company (a large furniture company) maintains an in-process inventory status database which contains the location and status of every part being processed in the plant. It is possible to download that database along with the MRP data (part routings, processing times, setup times, lot size, etc.) to a PC. In addition, material handling and manpower resources (e.g., quantity and assignment) are provided from other databases.

Once the entire data set is resident in the PC, it is automatically processed into the Virtual Factory model. The model is constructed as an object-oriented simulation using C++. From the in-process inventory database, instances of class objects are constructed and initialized. The class objects were chosen in a manner to represent the actual hierarchy at the factory. For example, a `Cutting` class is instantiated that will contain many instances of the `End Item` class. Each `End Item` class will contain many instances of the `Part` class. Each instance of the `Part` class will have a list of instances of the `Operation` class that specify all future processing requirements. Figure 2 is a conceptual diagram of this hierarchical structure. Once created, all instances of the part classes are assigned to the queues of the specified machine centers based on their slack priority.

Once all parts in the factory have been instantiated and assigned to queues, an end of service is scheduled on the event calendar for the first job in each queue. The present time is then updated to the first scheduled end of service. At this point, an end of service is scheduled for the next part in queue at the current machine center. The just-finished part is assigned a material handler and scheduled an arrival at its next required machine center. If the next required machine center is idle, an end of service is scheduled for the part on that machine. If a queue exists, the part will be assigned a priority in the queue based upon its updated slack value.

When all parts have finished, new $slack'$ priorities are calculated based upon the queuing times of

Table 1: Computational Results - Industrial Examples

Data Set	Jobs	Machines	$L_{\max}$ (hours)	LB($L_{\max}$) (hours)	#iterations to best $L_{\max}$	CPU Time (sec)
#1	485	81	6.78	6.78	1	0.550
#2	3237	81	151.53	151.53	3	13.650

the just completed simulation run. The system is then re-initialized with all parts being assigned to the queues of their first required machine center. However, the order of these parts is now based on the new revised slack ($slack'$) priority. The entire factory operation is then re-simulated, recording the queuing times of parts in the queues and re-calculating the $slack'$ quantity. The iterative procedure is concluded once the $L_{\max}$ value stops improving or the schedule converges.

3.2 Experimental Results

The iterative procedure was stopped if $L_{\max}$ = LB($L_{\max}$). For a description of the lower bound calculation, see (Hodgson et al. 1997). Otherwise, it was allowed to run for 50 iterations. As an initial test environment for the system, two data sets were used which were downloaded from the factory's MRP system. Data set #1 contains 485 jobs and 81 machines. Data set #2 contains 3237 jobs and 81 machines. (The current industrial data set has approximately 5800 parts and 225 machines and takes about 12 seconds on a 90 MHz Pentium PC.) In each case, jobs are initially in various states of completion, as would be the case in any factory. The procedure was run on each data set to determine the best $L_{\max}$, LB($L_{\max}$), and the procedure computation time. All computation was performed on a 66 MHz Pentium PC.

In order to more fully exercise the procedure, a number of randomly generated jobs were tested. Of the 120 problems generated, 88 (73.3%) were solved to the lower bound LB($L_{\max}$). A more detailed inspection of the results reveals that 97 problems (80.8%) were solved to within LB($L_{\max}$)+1.0, 104 problems (86.7%) were solved to within LB($L_{\max}$)+2.0, and 113 problems (94.2%) were solved to within LB($L_{\max}$)+4.0. This comparison is with the lower bound rather than a known optimal solution, with the average job processing time per machine being approximately 5.5. Considering the relatively slow computer used, the computation times are very good for the size problems involved.

If the number of iterations of the procedure had been limited to 10 (rather than 50), 83 problems (69.2%) would have been solved to LB($L_{\max}$). There would have been little deterioration of the solutions not reaching the lower bound, and there would have been reductions in the average CPU time of roughly 70% to 80%. Details of these results are reported in (Hodgson et al. 1997).

4 PRESENTATION OF STATISTICS AND MODEL INFORMATION

The industrial partner discussed in this paper has made the transition to cellular manufacturing for strategic reasons. Therefore, the statistics and model information are presented at three levels coinciding with this setup. The machine level, (the most detailed), includes a priority list of jobs for each machine. This priority list includes the part name, the part number, its previous operation (in case it is not at the correct machine center), and the sequence in which the parts should be processed at that machine.

The second level of information is the cell management level. Since the cell leader is primarily concerned with manpower and staffing, the equivalent man-hours of work in queue can be displayed. The utilization (over time) of each machine within a cell is also displayed alongside the hours in queue. Figure 3 shows these graphs for a typical machine. The graph on the left is the machine utilization over time. On the right is the equivalent hours in queue. This information can also be viewed simultaneously in chart form for all machines within a cell. The cell leader also has a priority list of all parts to come through the cell during a shift for all machines.

The highest level of information is the factory level. Summary statistics are aggregated by cell to help the scheduler determine which cells are under- utilized or over-utilized. In Figure 4, the equivalent man-hours is compared to the maximum possible capacity for the cell. The actual staffing is in the lower right hand corner.

Diagnostic statistics are aggregated over a long-range interval of the simulation and presented on a conceptual diagram of the factory, with each cell represented by a colored block. These cell blocks are color-coded red (problem area), yellow (caution)

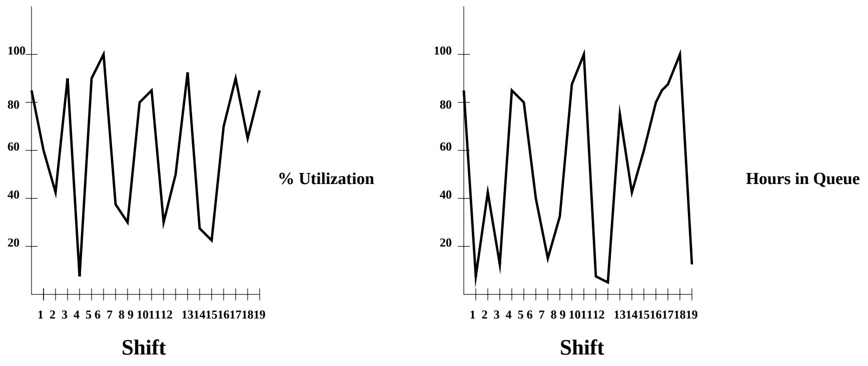
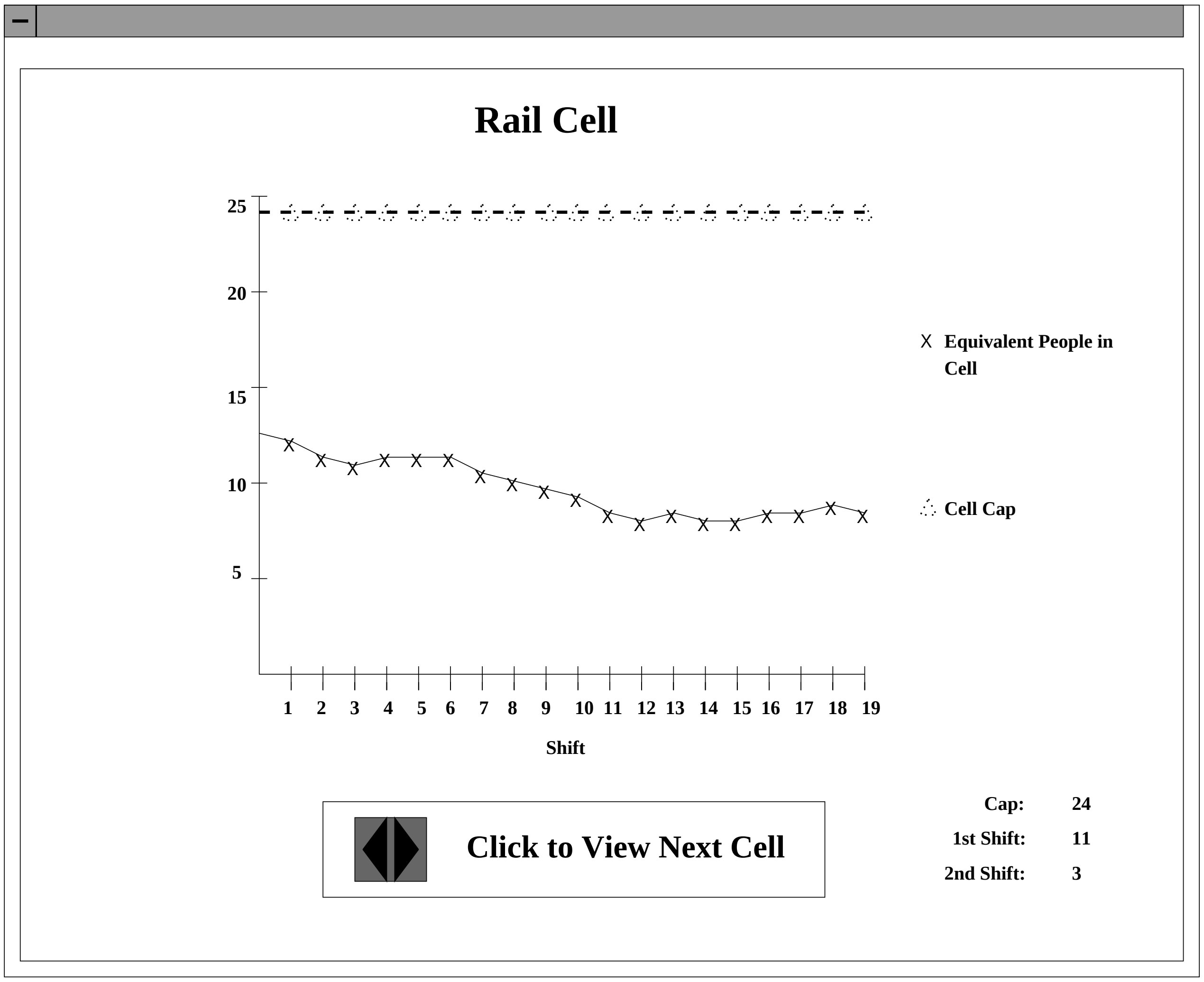


Figure 3: Utilization and Hours in Queue for a Typical Machine



Cap:	24
1st Shift:	11
2nd Shift:	3

Figure 4: Cell Level Comparison of Capacity and Hours in Queue

and green (acceptable) based on combinations of the statistics. Exactly which statistical combinations indicate a problem area will vary based on management preferences.

The emphasis of the statistics on this factory-level diagram is to help the scheduler determine why parts are late. Therefore, the ability to follow "hot" parts (ones which cause an end-item to be late) to find out why they are late is important.

The cell information includes: the cell name and number, the number of required equivalent people (the sum of the processing time multiplied by the number of people required to run the machines divided by shift length), the maximum machine utilization (and the machines name and identification number), and the number of parts which ended up tardy that went through that cell. Also given are the average and maximum times parts spend in queue for each cell.

5 CONCLUSION

This paper has described the use of object-oriented simulation as the driver behind a scheduling system application. Using the capabilities of object-oriented simulation allows for the model to be run quickly, increasing its usability for "what-if" scenarios. Since the simulation model is finite capacity, the schedules which are produced are realistic and can be attained on the shop floor. In addition, the detailed nature of simulation allows the user(s) to precisely capture statistics and generate job priority lists for several levels of management.

ACKNOWLEDGMENTS

This research was sponsored, in part, by Office of Naval Research Grant N00014-90-J-1045, and by the Furniture Manufacturing and Management Center at North Carolina State University.

REFERENCES

Carroll, D.C. 1965. Heuristic sequencing of single and multiple component jobs. Doctoral thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Hodgson, T.J., Cormier, D., Weintraub, A., and Zozom, A. 1997. A scheduling procedure to satisfy due dates in large job shops. Technical Report No. 97-01: Department of Industrial Engineering, North Carolina State University, Raleigh, North Carolina.

Morton, T.E., and Pentico, D.W. 1993. *Heuristic scheduling systems.* New York: John Wiley & Sons, Inc..

Plossl, G.W. 1985. *Production and inventory control.* 2nd ed. New Jersey: Prentice Hall.

Rowe, A. 1960. Towards a theory of scheduling *Industrial Engineering* 11 (1).

AUTHOR BIOGRAPHIES

ALEXANDER J WEINTRAUB is a Ph.D. Candidate in the Department of Industrial Engineering and a Research Assistant in Furniture Manufacturing and Management Center at North Carolina State University. He received a B.S. degree in mechanical engineering from Rensselaer Polytechnic Institute in 1991, and he received an M.S. degree in mechanical engineering from Columbia University in 1994. He is a member of Alpha Pi Mu, INFORMS, and IIE. His interests include production and manufacturing systems, scheduling, and object-oriented simulation.

ANDREW ZOZOM, JR. is a Ph.D. Candidate in the Department of Industrial Engineering and a Research Assistant in Furniture Manufacturing and Management Center at North Carolina State University. He received a B.S. degree in industrial engineering and operations research from Virginia Tech in 1990, and he received an M.S. degree in industrial engineering and operations research from North Carolina State University in 1996. He is a member of Alpha Pi Mu, INFORMS, IIE and APICS.

THOM J. HODGSON is Professor of Industrial Engineering and Director of the Integrated Manufacturing Systems Engineering Institute at North Carolina State University. He has served as a Division Director at the National Science Foundation and is a former Editor-in-Chief of the IIE Transactions. His research interests are in the area of production and inventory control, scheduling, and manufacturing systems.

DENIS CORMIER is an Assistant Professor of Industrial Engineering at North Carolina State University. His current research interests include computer aided process planning, rapid prototyping and tooling, and hierarchical control.